

Workshop on the ACTS Toolkit
September 28–30, 2000
National Energy Research Scientific Computing Center

TAO – Toolkit for Advanced Optimization

Steve Benson, Lois Curfman McInnes, and Jorge J. Moré

<http://www.mcs.anl.gov/tao>

Mathematics and Computer Science Division
Argonne National Laboratory



What is Nonlinearly Constrained Optimization?

$$\min \{ f(x) : x_l \leq x \leq x_u, c_l \leq c(x) \leq c_u \}$$

- ◊ Systems of nonlinear equations

$$\min \left\{ \frac{1}{2} \|r(x)\|^2 : x_l \leq x \leq x_u \right\}, \quad r : \mathbb{R}^n \mapsto \mathbb{R}^n$$

- ◊ Nonlinear least squares

$$\min \left\{ \frac{1}{2} \|r(x)\|^2 : x_l \leq x \leq x_u \right\}, \quad r : \mathbb{R}^n \mapsto \mathbb{R}^m, \quad m \geq n$$

What is Nonlinearly Constrained Optimization?

$$\min \{f(x) : x_l \leq x \leq x_u, c_l \leq c(x) \leq c_u\}$$

- ◊ Unconstrained optimization

$$\min \{f(x) : x \in \mathbb{R}^n\}$$

- ◊ Bound-constrained optimization

$$\min \{f(x) : x_l \leq x \leq x_u\}$$

- ◊ Linear and quadratic programming

$$\min \left\{ \frac{1}{2}x^T Qx + c^T x : x_l \leq x \leq x_u, c_l \leq Ax \leq c_u \right\}$$

The Ginzburg-Landau Model for Superconductivity

Minimize the Gibbs free energy for a homogeneous superconductor with a vector potential perpendicular to the superconductor.

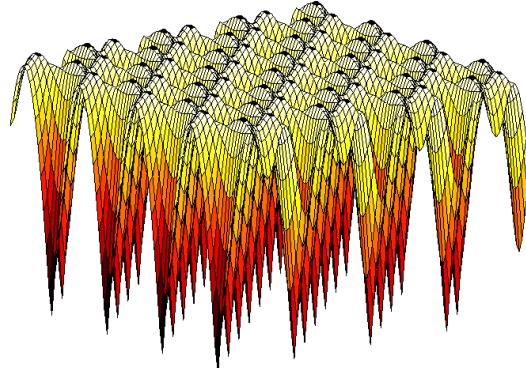
$$\int_{\mathcal{D}} \left\{ -|v(x)|^2 + \frac{1}{2}|v(x)|^4 + \|[\nabla - iA(x)] v(x)\|^2 + \kappa^2 \|(\nabla \times A)(x)\|^2 \right\} dx$$

$v : \mathbb{R}^2 \rightarrow \mathbb{C}$ is the order parameter

$A : \mathbb{R}^2 \rightarrow \mathbb{R}^2$ is the vector potential

The Ginzburg-Landau Model for Superconductivity

Unconstrained problem. Non-convex function. Hessian is singular.
Unique minimizer, but there is a saddle point.



Pressure Distribution in a Journal Bearing

Determine the pressure distribution in a thin film of lubricant between two circular cylinders

$$\int_{\mathcal{D}} \left\{ \frac{1}{2} w_q(x) \|\nabla v(x)\|^2 - w_l(x) v(x) \right\} dx$$

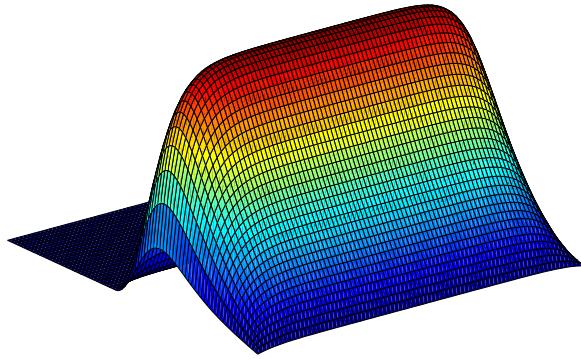
$$w_q(\xi_1, \xi_2) = (1 + \varepsilon \cos(\xi_1))^2$$

$$w_l(\xi_1, \xi_2) = \varepsilon \sin(\xi_1)$$

$v \geq 0$ on the domain $\mathcal{D} = (0, 2\pi) \times (0, 2b)$.

Pressure Distribution in a Journal Bearing

Bound constrained problem. Number of active constraints depends on the eccentricity ε . Badly scaled Hessian matrix.



Minimal Surface with Obstacles

Determine the surface of minimal area and given boundary data that lies above an obstacle.

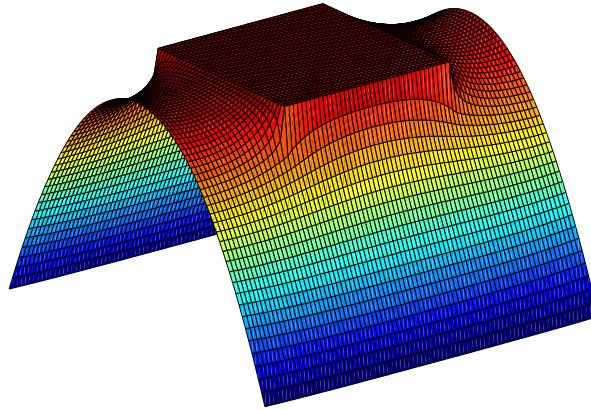
$$\min \{f(v) : v \in K\}$$

$$f(v) = \int_{\mathcal{D}} \sqrt{1 + \|\nabla v(x)\|^2} dx$$

$$K = \{v \in H^1 : v(x) = v_D(x), \quad x \in \partial D, \quad v(x) \geq v_L(x), \quad x \in \mathcal{D}\}$$

Minimal Surface with Obstacles

Bound constrained problem. Number of active constraints depends on the height of the obstacle. All multipliers are zero.



Isomerization of α -pinene

Determine the reaction coefficients in the thermal isomerization of α -pinene from measurements by minimizing

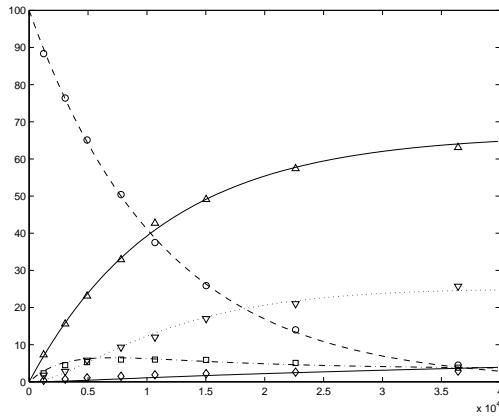
$$\sum_{j=1}^8 \|y(\tau_j; \theta) - z_j\|^2,$$

where z_j are the measurements and

$$\begin{aligned} y'_1 &= -(\theta_1 + \theta_2)y_1 \\ y'_2 &= \theta_1 y_1 \\ y'_3 &= \theta_2 y_1 - (\theta_3 + \theta_4)y_3 + \theta_5 y_5 \\ y'_4 &= \theta_3 y_3 \\ y'_5 &= \theta_4 y_3 - \theta_5 y_5 \end{aligned}$$

Isomerization of α -pinene

Only equality constraints. Typical parameter estimation problem.



Optimization Toolkits

State-of-the-art in optimization software:

- ◊ Scattered support for parallel computations
- ◊ Little reuse of linear algebra software
- ◊ Minimal use of automatic differentiation software
- ◊ Few object-oriented optimization codes
- ◊ Nonlinear optimization problems with more than 10,000 variables are considered large.

TAO

The process of nature by which all things change and which is to be followed for a life of harmony.

The Right Way

Toolkit for advanced optimization

- ◊ Object-oriented techniques
- ◊ Component-based interaction
- ◊ Leverage of existing parallel computing infrastructure
- ◊ Reuse of external toolkits

TAO Goals

- ◊ Portability
- ◊ Performance
- ◊ Scalable parallelism
- ◊ An interface independent of architecture

TAO Performance: Journal Bearing Problem (IBM SP)

$n = 2.56 \cdot 10^6$ variables

ε	p	faces	n_{CG}	time	$t_{CG}\%$	\mathcal{E}
0.1	8	46	431	7419	86	100
	16	45	423	3706	83	100
	32	45	427	2045	82	91
	64	45	427	1279	82	73
0.9	8	37	105	2134	70	100
	16	37	103	1124	71	95
	32	38	100	618	69	86
	64	38	99	397	68	67

TAO Algorithms (partial list)

- ◊ Unconstrained optimization
 - Conjugate gradient algorithms PR, FR, PR+
 - Levenberg-Marquardt method (alpha)
- ◊ Bound-constrained optimization
 - Limited-memory variable-metric algorithm
 - Trust region Newton method
- ◊ Linearly constrained optimization
 - Interior-point quadratic programming method (alpha)
- ◊ Nonlinearly constrained optimization
 - Work in progress

TAO Algorithms for Bound-Constrained Optimization

$$\min \{f(x) : x_l \leq x \leq x_u\}$$

- ◊ Conjugate gradient algorithms
- ◊ Limited-memory variable-metric algorithms
- ◊ Newton algorithms

You must supply the function $f : \mathbb{R}^n \mapsto \mathbb{R}$ and the gradient

$$\nabla f(x) = (\partial_i f(x))$$

For Newton methods you also need to supply the Hessian matrix.

$$\nabla^2 f(x) = (\partial_{i,j} f(x))$$

Conjugate Gradient Algorithms

$$x_{k+1} = x_k + \alpha_k p_k$$

$$p_{k+1} = -\nabla f(x_k) + \beta_k p_k$$

where α_k is determined by a line search.

Three choices of β_k are possible ($g_k = \nabla f(x_k)$):

$$\beta_k^{FR} = \left(\frac{\|g_{k+1}\|}{\|g_k\|} \right)^2, \quad \text{Fletcher-Reeves}$$

$$\beta_k^{PR} = \frac{\langle g_{k+1}, g_{k+1} - g_k \rangle}{\|g_k\|^2}, \quad \text{Polak-Rivière}$$

$$\beta_k^{PR+} = \max \{\beta_k^{PR}, 0\}, \quad \text{PR-plus}$$

Limited-Memory Variable-Metric Algorithms

$$x_{k+1} = x_k - \alpha_k H_k \nabla f(x_k)$$

where α_k is determined by a line search.

The matrix H_k is defined in terms of information gathered during the previous m iterations.

- ◊ H_k is positive definite.
- ◊ Storage of H_k requires $2mn$ locations.
- ◊ Computation of $H_k \nabla f(x_k)$ costs $(8m + 1)n$ flops.

Trust Region Newton Algorithm

At each iteration the step s_k (approximately) minimizes

$$\min \{q_k(x_k + s) : s_i = 0, i \in \mathcal{A}_k, x_l \leq x_k + s \leq x_u, \|s\| \leq \Delta_k\}$$

where q_k is the quadratic approximation,

$$q_k(w) = \langle \nabla f(x_k), w \rangle + \frac{1}{2} \langle w, \nabla^2 f(x_k) w \rangle,$$

to the function, and Δ_k is the trust region bound.

- ◊ Predict an active set \mathcal{A}_k .
- ◊ Compute a step s_k
- ◊ $x_{k+1} = x_k + s_k$ if $f(x_k + s_k) < f(x_k)$, otherwise $x_{k+1} = x_k$.
- ◊ Update Δ_k .

TAO Interface

```
TAO_SOLVER tao;           /* TAO_SOLVER solver context */
TaoMethod method='tao_lmvm'; /* A particular solver method to be used */
Vec      x, g;           /* solution and gradient vectors */
int      n;               /* number of variables */
AppCtx   user;            /* user-defined application context */
int      info;             /* status code */

VecCreate(MPI_COMM_WORLD,n,&x);
VecDuplicate(x,&g);

info = TaoCreate(MPI_COMM_WORLD,method,&tao);
info = TaoSetFunctionGradient(tao,x,g,FunctionGradient,(void *)&user);
info = TaoSolve(tao);

info = TaoDestroy(tao);
```

Function Evaluation

```
typedef struct          /* Used in the minimum surface area problem */
  int      mx, my;        /* discretization in x, y directions */
  Vec      Bottom, Top, Left, Right; /* boundary values */
AppCtx;

int FormFunction(TAO_SOLVER tao, Vec x, double* fcn,void *userCtx){
  AppCtx *user = (AppCtx *)userCtx;
  ...
  return 0;
}
```

The user sets this routine in the main program via

```
info = TaoSetFunction(tao,x,FormFunction,(void *)&user);
```

Gradient Evaluation

```
int FormGradient(TAO_SOLVER tao, Vec x, Vec g,void *userCtx){  
    AppCtx *user = (AppCtx *)userCtx;  
    ...  
    return 0;  
}
```

The user sets this routine in the main program via

```
info = TaoSetGradient(tao,g,FormGradient,(void *)&user);
```

Alternatively, the user can supply the function and gradient evaluation in a single routine.

A Hessian evaluation routine can be supplied in a similar manner.

Convergence

Absolute tolerances specify acceptable errors in the optimality of the function and the constraints.

$$f(x) \leq f(x^*) + \epsilon_{fatol}$$

Relative tolerances specify the number of significant digits required in the solution and the constraints.

$$f(x) \leq f(x^*) + \epsilon_{frtol}|f(x^*)|$$

These tolerance can be changed

```
int TaoSetTolerances(TAO_SOLVER solver,double fatol,double frtol,  
                     double catol,double crtol)
```

TAO Basic Facilities

- ◊ TaoInitialize
- ◊ TaoFinalize
- ◊ TaoSetInitialVector
- ◊ TaoSetBounds
- ◊ TaoGetSLES
- ◊ TaoGetFunctionValue
- ◊ TaoCheckSuccess
- ◊ TaoView

Parallel Functionality

The TAO interface is the same in a parallel environment, but the user must provide vectors with a parallel structure.

```
VecCreateMPI(MPI_COMM_WORLD,n,PETSC_DECIDE,&x);
VecDuplicate(x,&g);

info = TaoCreate(MPI_COMM_WORLD,method,&tao);
info = TaoSetFunctionGradient(tao,x,g,FunctionGradient,(void *)&user);
info = TaoSolve(tao);

info = TaoDestroy(tao);
```

The user still provides the routines that evaluate the function and gradient. These routines do not have to be performed in parallel, but parallel evaluations usually improve performance.

Parallel Function Evaluation

```
typedef struct {                                /* For Minimum Surface Area Problem */
    int          mx, my;                      /* discretization in x, y directions */
    Vec          Bottom, Top, Left, Right;      /* boundary values */
    Vec          localX, localG;                /* ghosted local vector */
    DA           da;                          /* distributed array data structure */
} AppCtx;

int FormFunction(TAO_SOLVER tao, Vec x, double* fcn, void *userCtx) {
    AppCtx *user = (AppCtx *)userCtx;
    ...
    info = MPI_Allreduce(&f, fcn, 1, MPI_DOUBLE, MPI_SUM, MPI_COMM_WORLD);
    return 0;
}
```

Where Can We Find TAO?

At <http://www.mcs.anl.gov/tao>

- ◊ Source Code
- ◊ Documentation
- ◊ Installation Instructions
- ◊ Example Problems
- ◊ Additional Tutorials
- ◊ Performance Results
- ◊ Supported Architectures